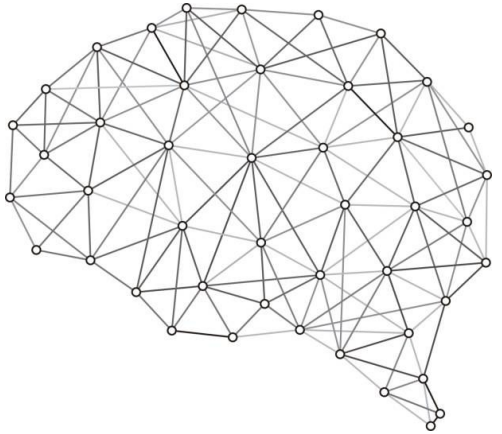


Aula 06 - ESP32 - Logger para sensor de movimento

Instituto Iracema
PESQUISA E INOVAÇÃO

Módulo de Internet das Coisas

- Prof.

UNIVERSIDADE ESTADUAL DO CEARÁ

MCTI FUTURO

Softex

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

GOVERNO FEDERAL
BRASIL
UNIÃO E RECONSTRUÇÃO

IA

Objetivos da Aula

- Introduzir o logger de monitoramento de sensor
- Acessar “Sistemas de Arquivos” em memória flash (SPIFFS)
- Armazenar dados de sensor em flash
- Construir um simples logger de monitoramento de sensor

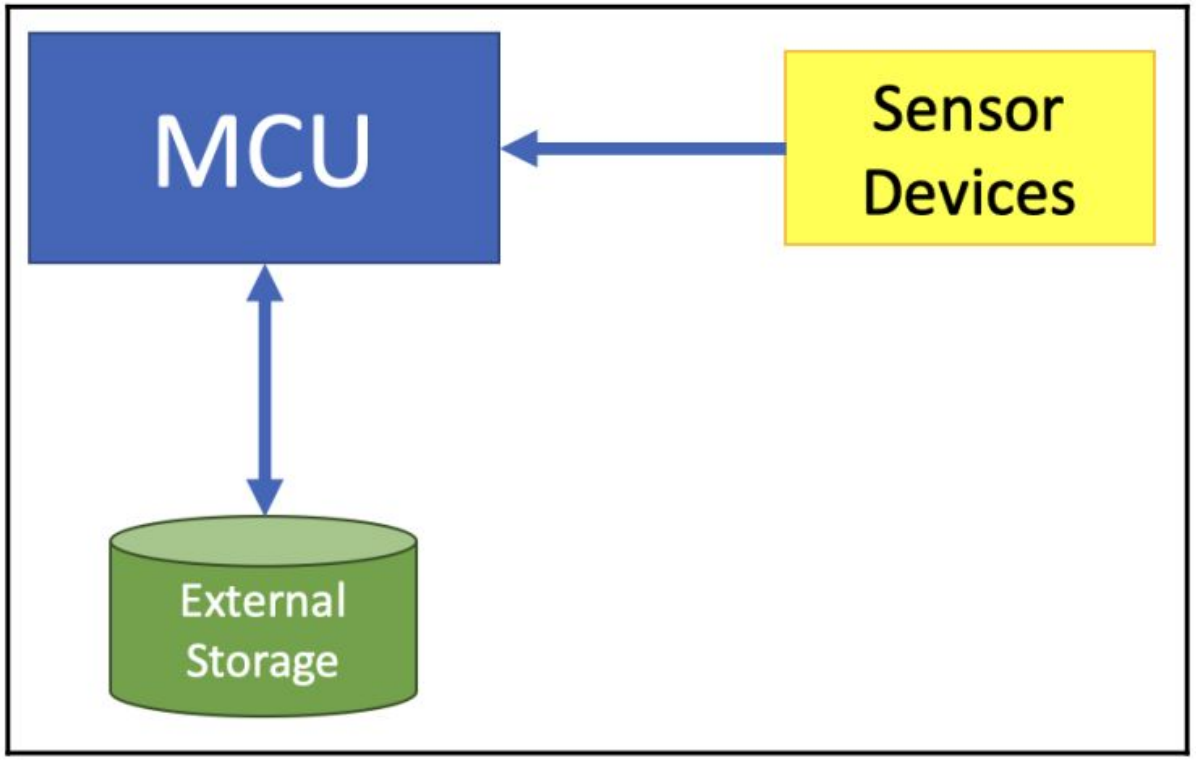
IA

Requisitos técnicos

- Um computador com um SO instalado como Windows, Linux ou macOS
- Uma placa de desenvolvimento ESP32 cam TTGO

Introdução ao logger de monitoramento de sensor [1]

- Um sistema de logging é um sistema que pode escrever dados e informações
  - ex. dados de sensor, eventos do sistema e mensagens de erro
- Todos dados e informações geralmente são armazenados em armazenamento externo como cartões SD, micro SD ou disco rígido.
- Mas nada impede que dados de sensores ou da própria aplicação sejam armazenados na memória local.



IA

Criar um log local na memória flash [2]

1. Usar o Arduino IDE-> Criar novo Sketch: LogMotionSensor
2. Verificar o particionamento da memória em Ferramentas -> PartitionScheme -> **Default**
3. Incluir bibliotecas FS.h e SPIFFS.h
4. Declarar o PIN do PIR Sensor e arquivo do log
5. Criar função setup() para inicializar do sensor, comunicação serial
6. Criar função logMotion () para registrar no arquivo o movimento detectado
7. Criar função readLogFile () para ler o arquivo de log
8. Criar função loop () de execução da aplicação

Criar um log local na memória flash

1. Usar o Arduino IDE-> Criar novo Sketch: LogMotionSensor
2. Incluir bibliotecas FS.h e SPIFFS.h

```
#include <FS.h>
#include <SPIFFS.h>
```

FS.h é uma biblioteca de uso geral para operações do sistema de arquivos no microcontrolador ESP32. Ele fornece uma interface para acessar os sistemas de arquivos SPIFFS e LittleFS, bem como cartões SD e outros tipos de dispositivos de armazenamento.

O SPIFFS.h, por outro lado, é uma biblioteca específica para acessar o sistema de arquivos SPIFFS (SPI Flash File System) no ESP32. O SPIFFS é um sistema de arquivos leve otimizado para uso em dispositivos incorporados com recursos limitados.

IA

Criar um log local na memória flash

3. Declarar o PIN do PIR Sensor e arquivo do log

```
const int PIR_PIN = 33;
const char* FILE_NAME = "/motion_log.txt";
```

IA

Criar um log local na memória flash

4. Criar função setup() para inicializar do sensor, comunicação serial

```
void setup() {
  pinMode(PIR_PIN, INPUT);
  Serial.begin(115200);
  if(!SPIFFS.begin(true)){
    Serial.println("An error occurred while mounting SPIFFS");
    return;
  }
  Serial.println("Type '1' to read the log ");
}
```

pinMode() : Define o “PIR_PIN” (pino 33, correspondente ao sensor PIR) como um input. O primeiro argumento da função é o número do pino e o segundo argumento especifica se o pino deve ser configurado como entrada ou saída.

Serial.begin(115200), configura a comunicação serial com uma taxa de transmissão de 115200 bits por segundo. Essa função é usada para inicializar a comunicação serial entre o microcontrolador e outro dispositivo, como um computador ou outro microcontrolador.

SPIFFS.begin(true), inicia o sistema de arquivos SPIFFS (SPI Flash File System) na placa, permitindo possa armazenar e ler arquivos no chip de memória flash da placa.O argumento true passado para a função indica que o sistema de arquivos deve ser formatado antes de ser montado. Isso significa que, se houver algum dado no SPIFFS, ele será apagado e o sistema de arquivos será reformatado.

IA

Criar um log local na memória flash

5. Criar função logMotion () para registrar no arquivo o movimento detectado

```
void logMotion() {
  Serial.println("Motion detected!");
  Serial.println("Type '1' to read the log ");
  File logFile = SPIFFS.open(FILE_NAME, FILE_APPEND);
  if (!logFile) {
    Serial.println("Failed to open log file");
    return;
  }
  logFile.println("Motion detected!");
  logFile.println("-----------------------");
  logFile.close();
}
```

SPIFFIS.open(FILE_NAME, FILE_APPEND) : É uma chamada que abre um arquivo existente ou cria um novo arquivo com o nome especificado em FILE_NAME, que é uma constante que contém o caminho e o nome do arquivo que você deseja manipular. O segundo argumento FILE_APPEND indica que o arquivo deve ser aberto em modo de escrita com o ponteiro de escrita posicionado no final do arquivo. Isso significa que, quando você escreve no arquivo, os dados são adicionados ao final do arquivo, em vez de substituir o conteúdo existente.

IA

Criar um log local na memória flash

6. Criar função readLogFile () para ler o arquivo de log.

Se o arquivo for aberto com sucesso, a função entra em um loop while que verifica se ainda há conteúdo disponível no arquivo para leitura. Se houver conteúdo disponível, a função lê e exibe o conteúdo na porta serial usando a função Serial.write(). Quando todo o conteúdo do arquivo tiver sido lido, a função logFile.close() é chamada para fechar o arquivo e liberar os recursos do sistema.

```
void readLogFile() {
  Serial.println("Reading log file:");
  File logFile = SPIFFS.open(FILE_NAME);
  if (!logFile) {
    Serial.println("Failed to open log file");
    return;
  }
  while (logFile.available()) {
    Serial.write(logFile.read());
  }
  logFile.close();
}
```

IA

Criar um log local na memória flash

7. Criar função loop () de execução da aplicação

Dentro da função loop(), a primeira coisa que é feita é ler o estado do pino do sensor PIR usando a função digitalRead(PIR_PIN). Se o valor lido for HIGH (ou seja, detectado movimento pelo sensor), a função logMotion() será chamada para registrar o evento em um arquivo de log.

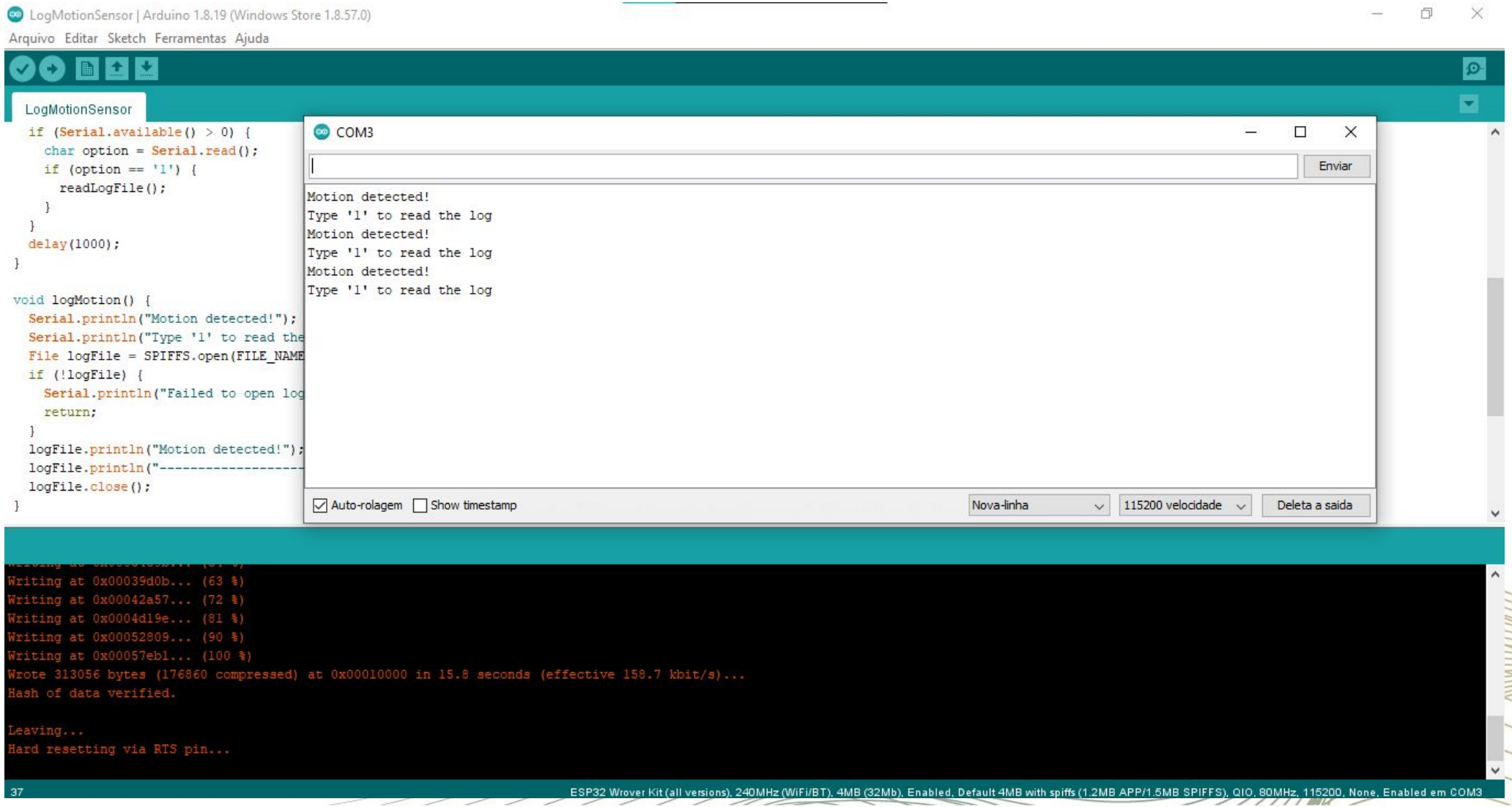
Em seguida, o código verifica se há dados disponíveis para leitura na porta serial usando a função Serial.available(). Se houver dados disponíveis, o código lê o primeiro byte disponível usando a função Serial.read() e armazena em uma variável chamada option. Se o valor armazenado na variável option for igual a 1, a função readLogFile() será chamada para ler e exibir o conteúdo do arquivo de log.

Finalmente, a função delay(1000) é usada para fazer o programa esperar por 1 segundo antes de reiniciar o loop.

```
void loop() {
  int motion = digitalRead(PIR_PIN);
  if (motion == HIGH) {
    logMotion();
  }
  if (Serial.available() > 0) {
    char option = Serial.read();
    if (option == '1') {
      readLogFile();
    }
  }
  delay(1000);
}
```

IA

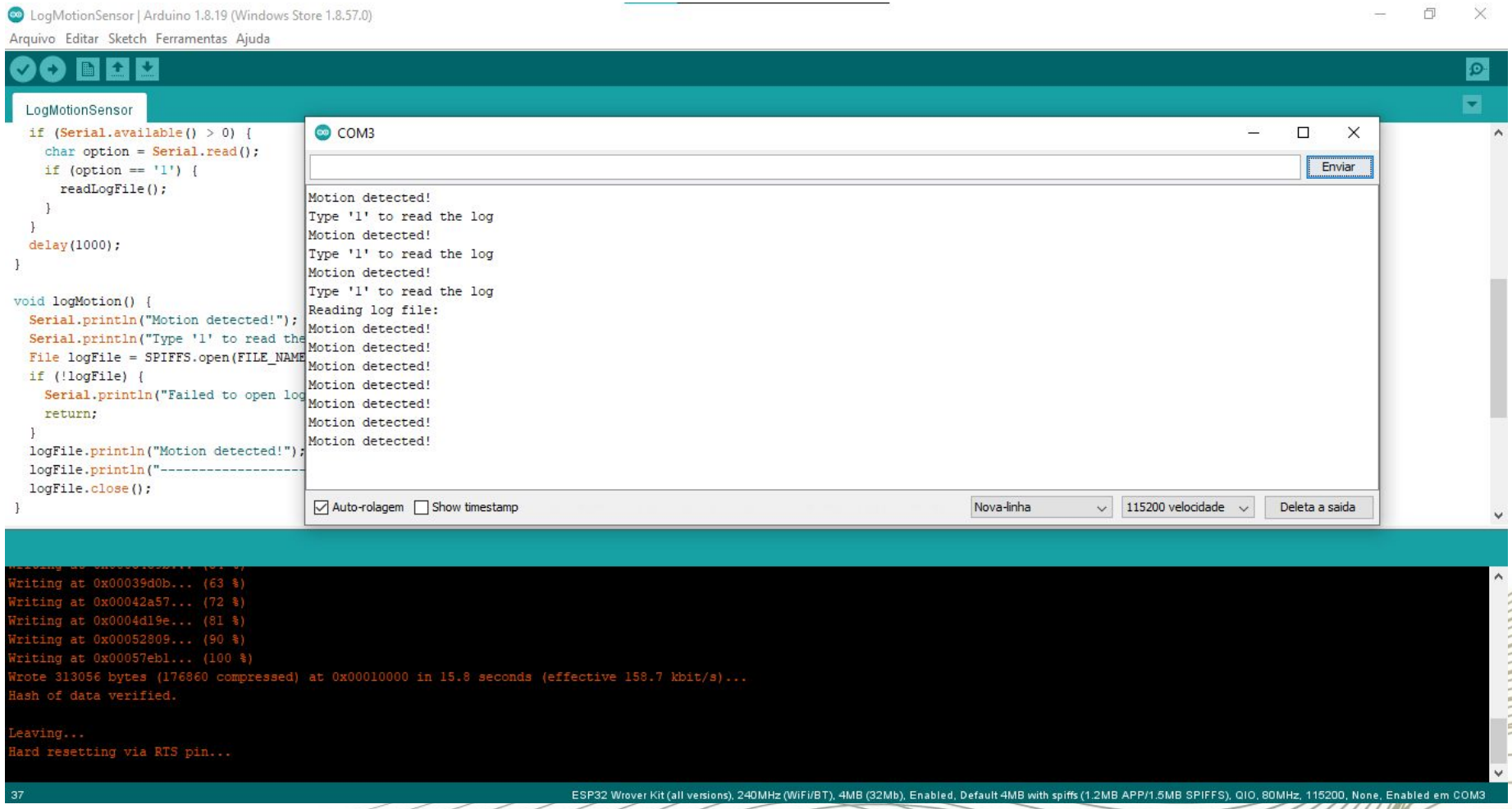
Execução do logger



IA

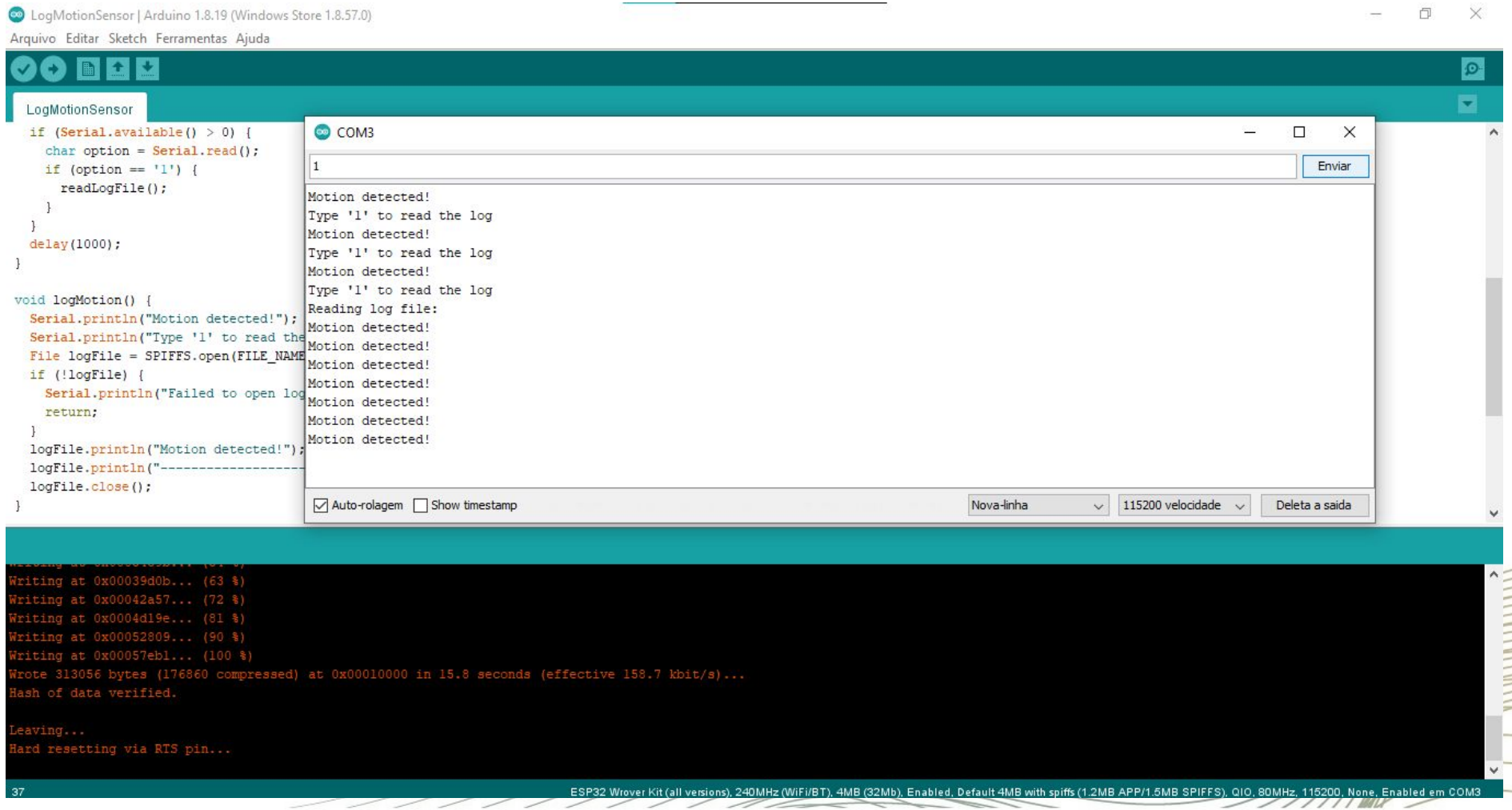
Execução do logger

Movimento detectado e registro no arquivo



Execução do logger

Exibição de conteúdo do log



IA

Referência Bibliográfica

[1] Kurniawan, A. Internet of Things Projects with ESP32. 2019. Packt Publishing.

[2]Rui Santos. ESP32 Flash Memory-Store Permanent Data (Write and Read). 2018. Disponível em: <https://randomnerdtutorials.com/esp32-flash-memory/>. Acesso em: 21/03/2023.

Dúvidas

?

Módulo de Intenet das Coisas